Rigorous Approximated Determinization of Weighted Automata

Benjamin Aminof*, Orna Kupferman*, and Robby Lampert†
*Hebrew University, Israel †Weizmann Institute, Israel
Email: {benj,orna}@cs.huji.ac.il robby.lampert@weizmann.ac.il

***Abstract*—A nondeterministic weighted finite automaton (WFA) maps an input word to a numerical value. Applications of weighted automata include formal verification of quantitative properties, as well as text, speech, and image processing. Many of these applications require the WFAs to be deterministic, or work substantially better when the WFAs are deterministic. Unlike NFAs, which can always be determinized, not all WFAs have an equivalent deterministic weighted automaton (DWFA). In [1], Mohri describes a determinization construction for a subclass of WFA. He also describes a property of WFAs (the *twins property*), such that all WFAs that satisfy the twins property are determinizable and the algorithm terminates on them. Unfortunately, many natural WFAs cannot be determinized.**

In this paper we study *approximated determinization* of WFAs. We describe an algorithm that, given a WFA $\mathcal{A}$ and an approximation factor $t \geq 1$, constructs a DWFA $\mathcal{A}'$ that t-*determinizes* $\mathcal{A}$. Formally, for all words $w \in \Sigma^*$, the value of w in $\mathcal{A}'$ is at least its value in $\mathcal{A}$ and at most t times its value in $\mathcal{A}$. Our construction involves two new ideas: attributing states in the subset construction by both upper and lower residues, and collapsing attributed subsets whose residues can be tightened. The larger the approximation factor is, the more attributed subsets we can collapse. Thus, t-determinization is helpful not only for WFAs that cannot be determinized, but also in cases determinization is possible but results in automata that are too big to handle. In addition, t-determinization is useful for reasoning about the competitive ratio of online algorithms. We also describe a property (the t-*twins property*) and use it in order to characterize t-determinizable WFAs. Finally, we describe a polynomial algorithm for deciding whether a given WFA has the t-twins property.

***Index Terms*—Weighted automata; Determinization;**

I. Introduction

Automata are the key to the modeling and solution of various problems in computer science. By reducing problems to questions about nondeterministic finite automata (NFAs, for short), we separate the algorithmic aspects of the problem, yielding clean and optimal solutions. For example, discrete feasible planning is reduced to the nonemptiness problem for NFAs [2], pattern finding in strings is reduced to the membership problem for NFAs [3], and correctness of finite-state systems with respect to safety properties is reduced to the containment problem for NFAs [4]. Research includes both efforts to find or improve automata-based frameworks for various settings, as well as a study of classical automata-theory problems, like the emptiness, membership, and containment problems mentioned above.

Over the years, researchers have extended the basic model of NFA, giving rise to automata-based frameworks for new settings. For example, in [5], Büchi introduced nondeterministic automata on infinite words, and used them in order to solve the decidability of S1S. Another example, which is the subject of this paper, is a generalization of NFAs to a multi-valued setting. While a classical NFA defines a subset of Σ^*, and hence maps each word in Σ^* to either 0 or 1, a *weighted finite automaton* (WFA, for short) maps each word in Σ^* to a value from some semiring [6], [1]. We focus on the *tropical* semiring $\langle \mathbb{R}^{\geq 0} \cup \{\infty\}, \min, +, \infty, 0 \rangle$. There, each transition of the WFA has a *weight* in $\mathbb{R}^{\geq 0}$, and the cost of a run is the sum of the weights of the transitions taken along the run. Applications of weighted automata over the tropical semiring include formal verification, where WFAs are used for the verification of quantitative properties [7], [8], for reasoning about probabilistic systems [9], and for reasoning about the competitive ratio of on-line algorithms [10], as well as text, speech, and image processing, where the weights of the WFA are used in order to account for the variability of the data and to rank alternative hypotheses [11], [12].

An NFA is nondeterministic, and may have several runs on an input word. In the Boolean setting, a word is accepted if some run accepts it. In the weighted setting, the cost that a WFA $\mathcal{A}$ assigns to a word w, denoted $cost(\mathcal{A}, w)$, is the minimum of the costs of accepting runs on w. For example, the WFA in Figure 1 has two accepting runs on the word abb. The first run is $q_0q_1q_1q_1$, and it has cost $1 + 1 + 1 = 3$. The second run is $q_0q_2q_2q_2$, and it has cost $2 + 2 + 2 = 6$. Thus, the cost that $\mathcal{A}$ assigns to abb is $\min\{3, 6\} = 3$.

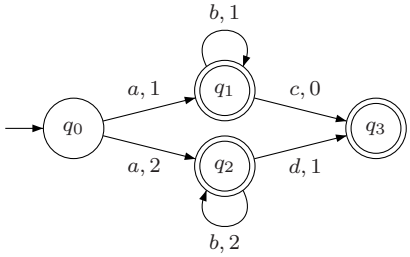

Fig. 1. A non-determinizable WFA.

As argued in [7], [8], [1], some applications of the automata-theoretic approach require, or work substantially better, when the automata are deterministic. In the context of formal verification, an implementation is correct with respect to its specification if the language of an automaton $\mathcal{I}$ that models the implementation is contained in the language of an automaton $\mathcal{S}$ that models the specification. In the weighted setting, a

solution to the containment problem is known only when $\mathcal{S}$ is deterministic [8][1], and the problem is in fact undecidable for WFAs [13]. Likewise, a translation of weighted automata to weighted μ-calculus, which is essential for symbolic algorithms, involves determinization of the automata, and so does the use of weighted automata as specifiers of winning conditions in weighted games [7]. In the context of speech recognition, weighted automata are used in order to represent components of a complex system, and the efficiency of combining the components crucially depends on the automata beeing deterministic [1], [14]. Another advantage of deterministic weighted automata (DWFAs, for short), is the existence of minimization algorithms for them [1]. In fact, while determinization (when possible) of WFA may involve a poly-exponential blow-up in the number of states, in practice determinization is remarkably successful, and DWFAs are not bigger than their nondeterministic origins [15].

Unlike NFAs, which can always be determinized [16], not all WFAs can be determinized. Consider for example the WFA from Figure 1. In order to get convinced that $\mathcal{A}$ does not have an equivalent DWFA, consider words of the form ab^*. It is not hard to see that an equivalent DWFA should reach different states after reading ab^i and ab^j, for $i \neq j$. Indeed, since $\mathcal{A}$ may read both c and d after reading a prefix in ab^*, and reading d forces the cost accumulated so far to be doubled, a DWFA for the language must remember this unbounded accumulated cost.

In [1], Mohri describes a determinization construction for a subclass of WFA. Essentially, as in the subset construction for NFA [16], each state in the equivalent DWFA is associated with a set S of states of the WFA. Intuitively, the weights on the transitions of the DWFA are defined so that the cost of reading a word w and getting to state S of the DWFA is equivalent to the minimal cost of reading w in the WFA and getting to some state in S. In order to achieve this, each state q in S is mapped to a *residue* – a value in $\mathbb{R}^{\geq 0}$ that describes the extra cost that has to be paid when the transition from S originates from a transition from q. The challenge in the determinization process is that these residues may keep increasing, bringing in more and more states associated with the set S, and the algorithm may not terminate even for WFAs that are determinizable.

Mohri also describes a property of WFAs – the *twins property*, such that all WFAs that satisfy the twins property are determinizable and the algorithm terminates on them. A WFA satisfies the twins property if for all pairs q and q' of states, if there are two words $u, v \in \Sigma^*$ such that both q and q' are reachable from the set of initial states along u, and both q and q' can loop along v, then the cost of looping along v from q is equal to the cost of looping along v from q'. The twins property captures a significant subclass of determinizable WFAs. In particular, for automata that are trim and unambiguous (all states participate in at least one accepting run, and each word accepted by the automaton has exactly one accepting run), satisfying the twins property is both a necessary and a sufficient condition for determinizability [1]. Moreover, checking whether a given unambiguous WFA satisfies the twins property can be done in polynomial time [17], [14]. We note that the problem of deciding whether a general WFA is determinizable is open. Also, a refined characterization of the settings in which Mohri's algorithm terminates in described in [18].

The need to work with DWFAs has called for improved solutions to the determinization challenge. One approach, taken in [7], is to extend weighted automata with registers that can maintain unbounded values. While this makes all automata determinizable, basic questions about the automata of [7] are undecidable, and decidability is obtained by augmenting the automaton with bound functions, which depends on the context in which the automaton is used. Another approach is to have an *approximated* determinization algorithm. Motivated by applications in speech recognition, [15] suggests a variant of Mohri's algorithm that allows the residues maintained in the state space of the DWFA to be approximated by some parameter ϵ. Thus, if during the subset construction we generate a state s that is ϵ-close to a state s' that has already been generated (formally, s and s' are associated with the same set S of states and the residues to which s maps the states in S are different by at most ϵ from those to which they are mapped in s'), then we give up the generation of s and use instead the state s'. Since the approximation mechanism in [15] is local, there is no way to relate $cost(\mathcal{A}, w)$ with $cost(\mathcal{A}', w)$, for a WFA $\mathcal{A}$ and its approximating DWFA $\mathcal{A}'$. Indeed, the only guarantee in the approximated-determinization construction of [15] is that $\mathcal{A}'$ accepts exactly all the words accepted by $\mathcal{A}$, and no guarantee is given about the cost of the accepted words. Nevertheless, the experimental results in [15] show that the approximation has led to a significant size reduction while hardly affecting the performance. In fact, for the application of speech recognition, researchers have tried even rougher approximations, like ignoring the weights of the WFA, and then re-introducing them via ad-hoc heuristics [19], [20]. A different approach to cope with the lack of a deteminization construction is to restrict attention to DWFAs that are embodied in the structure of the WFA [10].

The sequence of work above suggests that determinization of weighted automata is of great theoretical and practical interest, and that the lack of a rigorous approximated determinization construction should be addressed. By "rigorous" we mean that there is a guaranteed relation between the cost of words in the input WFA and the constructed DWFA. In this paper we solve this problem: we describe a rigorous approximated-determinization construction, and study its properties and applications. Given a WFA $\mathcal{A}$ and a real-valued parameter $t \geq 1$, we construct a DWFA $\mathcal{A}'$ that *t-approximates* $\mathcal{A}$. That is, for every word $w \in \Sigma^*$, the DWFA $\mathcal{A}'$ accepts w iff $\mathcal{A}$ accepts w, and $cost(\mathcal{A}, w) \leq cost(\mathcal{A}', w) \leq t \cdot cost(\mathcal{A}, w)$. We refer to such a construction as *t-determinization*. For example, in Figure 2 we describe a DWFA that 2-approximates

[1]In the weighted setting, a WFA $\mathcal{I}$ is contained in a WFA $\mathcal{S}$ iff for every word w, the cost that $\mathcal{I}$ assigns to w is less than, or equal to, the cost $\mathcal{S}$ assigns to w.

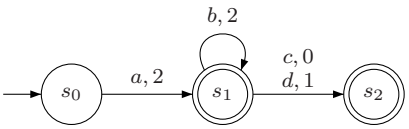

Fig. 2. A 2-determinization of the WFA from Figure 1.

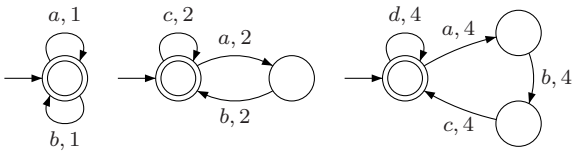

Fig. 3. A WFA that satisfies the 2-twins property.

the non-determinizable WFA in Figure 1.

A promising attempt to adjust Mohri's construction to t-determinization is to multiply the weights of some transitions of the given WFA by a factor of at most t, hoping to obtain a WFA that satisfies the twins property. Such an approach, however, ignores the subtle connection between different cycles of the WFA. To see the problem, consider the WFA appearing in Figure 3. While the WFA is 2-determinizable, we cannot multiply the weights of the transitions by a factor of at most 2 so that the result satisfies the twins property [21]. Indeed, since both the first and second components can traverse $(ab)^*$, and both the second and third components can traverse $(abc)^*$, no multiplication works.

The WFA appearing in Figure 3 also demonstrates that an adjustment of Mohri's construction to t-determinization by multiplying the weights of some transitions in the constructed DWFA by at most t, and updating the residues maintained in the states not to take into account "debts" that go below the multiplied weight, are doomed to fail too. While this construction, in case it terminates, results in a DWFA that t-approximates the input WFA, there are simple examples of WFAs (in particular, the WFAs in Figures 1 and 3) that are t-determinizable, yet the construction does not terminate on them. Indeed, a key point in the algorithm should be a mechanism for collapsing states associated with the same subset of states of the WFA to a single state. In Mohri's algorithm, when $t = 1$, the mechanism of maintaining residues proves itself as a very good one, and indeed the algorithm handles successfully all WFAs that satisfy the twins property. A good t-determinization algorithm should aim at similar high standards, which requires the development of a new collapsing mechanism.

Our t-determinization construction involves such a new mechanism, and indeed, as we prove, our construction terminates when applied to WFAs with rational weights that satisfy the *t-twins property*. Essentially, the t-twins property adds a parameter to Mohri's twins property and bounds by t the ratio between the costs of traversing cycles that can be traversed reading the same word. Note that the two WFAs in Figures 1 and 3 satisfy the 2-twins property. As has been the case with the twins property, the t-twins property captures a significant subclass of t-determinizable WFAs. In particular, as we show, for automata that are trim and unambiguous, satisfying the t-twins property is both a necessary and sufficient condition for t-determinizability. In addition, we present a polynomial-time algorithm for deciding whether an unambiguous WFA has the t-twins property.

The mechanism we suggest involves the following new ideas: Consider a WFA $\mathcal{A}$. Recall that each state p in the subset construction of $\mathcal{A}$ is associated with a set S of states of $\mathcal{A}$. We maintain for each state $q \in S$ two residues: an upper-bound residue u_q and a lower-bound residue l_q. The upper-bound residue plays a role similar to the one played by the single residue in Mohri's construction, and it upper bounds the cost that may be added to the cost of a run that proceeds from q without causing the cost of the run to exceed $t{\cdot}cost(\mathcal{A}, w)$. The lower-bound residue is a new feature and is the cost that should be added to the cost of a run that proceeds from q in order to make sure that the cost of the run is at least $cost(\mathcal{A}, w)$. Thus, each state $q \in S$ is associated with a range $[l_q, u_q]$ rather than with a single residue. This range is used in the criterion for collapsing states. Since the cost assigned by $\mathcal{A}'$ to a word w should be between $cost(\mathcal{A}, w)$ and $t{\cdot}cost(\mathcal{A}, w)$, the invariants maintained for u_q and l_q guarantee that every residual weight in the range $[l_q, u_q]$ may be used when we proceed from q without violating the approximation. Consequently, in case the algorithm is about to create a new state p' that corresponds to S and the algorithm has already generated a state p associated with S such that for every $q \in S$ the residual range of q in p' is contained in the residual range of q in p, then the algorithm does not create p', and uses the state p instead.

Our results enable t-determinization of automata for which determinization is impossible or not known. Recall that in some applications the user can settle for approximated determinization. Our approximated determinization may be useful even when the automata are determinizable. Indeed, we show that for all $t > t'$ there exists a WFA $\mathcal{A}$ such that a DWFA that t-approximates $\mathcal{A}$ is exponentially more succinct than one that t'-approximates $\mathcal{A}$. Finally, as we discuss in Section V, t-determinization has proven useful in an automata-theoretic approach for the competitive analysis of *online algorithms* [10], and our results here increase the domain of algorithms that can be handled by the framework.

II. Preliminaries

While standard automata map words in Σ^* to either "accept" or "reject", weighted automata may be viewed as partial functions (defined only for accepted words) from Σ^* to $\mathbb{R}^{\geq 0}$ (the set of non-negative reals). Formally, a *weighted finite automaton* (WFA, for short) is a 8-tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f \rangle$, where Σ is a finite input alphabet, Q is a finite set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, $c : \Delta \to \mathbb{R}^{\geq 0}$ is a weight function, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, $i : Q_0 \to \mathbb{R}^{\geq 0}$ is an initial-weight function, and $f : F \to \mathbb{R}^{\geq 0}$ is a final-weight function. A transition $d = \langle q, a, p \rangle \in \Delta$ (also written as $\Delta(q, a, p)$) can be taken by $\mathcal{A}$ when reading the input letter a in the state q, and it causes $\mathcal{A}$ to move to the state p with *cost* $c(d)$. The transition relation Δ induces a transition function $\delta : Q \times \Sigma \to 2^Q$ in the

expected way. Thus, for a state $q \in Q$ and a letter $a \in \Sigma$, we have $\delta(q,a) = \{p : \Delta(q,a,p)\}$. We extend δ to sets of states, by letting $\delta(S,a) = \bigcup_{q \in S} \delta(q,a)$, and recursively to words in Σ^*, by letting $\delta(S,\varepsilon) = S$, and $\delta(S, u \cdot a) = \delta(\delta(S,u),a)$, for all $u \in \Sigma^*$ and $a \in \Sigma$. A WFA $\mathcal{A}$ may be nondeterministic in the sense that it may have many initial states, and that for some $q \in Q$ and $a \in \Sigma$, it may have $\Delta(q,a,p_1)$ and $\Delta(q,a,p_2)$, with $p_1 \neq p_2$. If $|Q_0| = 1$ and for every state $q \in Q$ and letter $a \in \Sigma$ we have $|\delta(q,a)| \leq 1$ then $\mathcal{A}$ is a *deterministic* weighted finite automaton (DWFA, for short).

For a word $w = w_1 \dots w_n \in \Sigma^*$, and states $q, q' \in Q$, a *partial run* of $\mathcal{A}$ on w from q to q' is a sequence $r = r_0 r_1 \dots r_n \in Q^+$, where $r_0 = q, r_n = q'$, and for all $1 \leq i \leq n$, we have $d_i = \langle r_{i-1}, w_i, r_i \rangle \in \Delta$. The cost of the partial run r is $c(r) = \sum_{i=1}^{n} c(d_i)$. Note that if $\mathcal{A}$ is nondeterministic, it may have several partial runs on w from q to q'. The *partial cost* of w from q to q' in $\mathcal{A}$ is $\theta(q,w,q') = min\{c(r) : r \text{ is a partial run on } w \text{ from } q \text{ to } q'\}$. A *run* of $\mathcal{A}$ on a word $w \in \Sigma^*$ is a partial run $r = r_0 r_1 \dots r_n \in Q^+$ of $\mathcal{A}$ on w, where $r_0 \in Q_0$. The run r is *accepting* if $r_n \in F$. The word w is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The (unweighted) *language* of $\mathcal{A}$ is $L(\mathcal{A}) = \{w : w \text{ is accepted by } \mathcal{A}\}$. The cost of an accepting run is the sum of the weights of the transitions that participate in the run added to the initial weight of the first state and the final weight of the last state[2]. Formally, let $r = r_0 r_1 \dots r_n$ be an accepting run of $\mathcal{A}$ on w. The cost of r is $cost(\mathcal{A}, r) = i(r_0) + c(r) + f(r_n)$. The cost of w, denoted $cost(\mathcal{A}, w)$, is the minimal cost over all accepting runs of $\mathcal{A}$ on w. Thus, $cost(\mathcal{A}, w) = \min\{cost(\mathcal{A}, r) : r \text{ is an accepting run of } \mathcal{A} \text{ on } w\}$. For completeness, if $w \notin L(\mathcal{A})$ we set $cost(\mathcal{A}, w) = \infty$.

For two WFAs $\mathcal{A}$ and $\mathcal{A}'$, and an *approximation factor* $t \in \mathbb{R}$, $t \geq 1$, we say that $\mathcal{A}'$ *t-approximates* $\mathcal{A}$ iff for all words $w \in \Sigma^*$, we have $cost(\mathcal{A}, w) \leq cost(\mathcal{A}', w) \leq t \cdot cost(\mathcal{A}, w)$. We say that two weighted automata are *equivalent* if they accept the same set of words, with the same costs (equivalently, they 1-approximate each other).

A WFA is *trim* if every state appears in an accepting run on some word. A WFA is *unambiguous* (or *single-run*) if there exists exactly one accepting run for each accepted word. Consider a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f \rangle$. Two states, p and q, of $\mathcal{A}$ are *twins* if for all $u, v \in \Sigma^*$ such that $p, q \in \delta(Q_0, u)$, $p \in \delta(p, v)$, and $q \in \delta(q, v)$, it holds that $\theta(p,v,p) = \theta(q,v,q)$. The WFA $\mathcal{A}$ has the *twins property* if all pairs $p, q \in Q$ are twins.

The WFA in Figure 1, for example[3], is trim, as all its states appear in some accepting run and no transition weights ∞. It is, however, ambiguous, since it has two accepting runs on words of the form ab^*. In addition, it does not satisfy the twins property: the states q_1, q_2 are both reachable from the initial state by the word a, are both reachable from themselves by the word b, yet the costs of the two b-cycles are different.

[2]In general, a WFA may be defined with respect to any semiring $\langle \mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1} \rangle$. The cost of a run is then the semiring product of the initial weight of the first state, the weights along the run, and the final weight of the last state. The cost of an accepted word is the semiring sum over the costs of all accepting runs on it. In this work, we focus on weighted automata defined with respect to the *min-sum semiring*, $\langle \mathbb{R}^{\geq 0} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ (sometimes called the *tropical* semiring), as defined above.

[3]For convenience, throughout this paper, we set the values of the functions i and f to be constantly 0 and we omit them in the graphical descriptions.

III. Approximated Determinization

We describe an algorithm that given a WFA $\mathcal{A}$ and an approximation factor $t \geq 1$, constructs a DWFA $\mathcal{A}'$ that t-approximates $\mathcal{A}$. We say that $\mathcal{A}'$ is a *t-detrminization* of $\mathcal{A}$. Recall that not all WFAs are determinizable. In general, given t, not all WFAs are t-determinizable, and some WFAs are not t-determinizable for all $t \geq 1$. In Section IV we discuss t-determinizability and the class of WFAs that our algorithm t-determinizes. As discussed in Section I, approximated determinization may be applied also to determinizable automata, aiming at reducing the state space. Formally, we have the following.

Theorem 1. *For every $n \geq 1$ and approximation factor $t > 1$, there exists a WFA $\mathcal{A}_n^t$ with $O(n)$ states such that a DWFA that t-approximates $\mathcal{A}_n^t$ needs only two states whereas every DWFA that t'-approximates $\mathcal{A}_n^t$, for all $t' < t$, needs at least 2^n states.*

Proof: Let $\mathcal{A}_n^t$ be an automaton that accepts all words in $L_n = (a+b)^* \cdot a \cdot (a+b)^{n-1}$ with a cost of 1, and all words in $\{a,b\}^+ \setminus L_n$ with a cost of t.

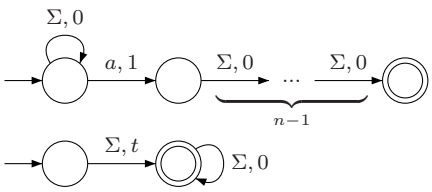

Fig. 4. The WFA $\mathcal{A}_n^t$.

It is easy to see that one can t-approximate $\mathcal{A}_n^t$ using a DWFA with two states that accepts all words in $\{a,b\}^+$ with a cost of t (the bottom component of $\mathcal{A}_n^t$). Since $\mathcal{A}_n^t$ satisfies the twins-property, then, by [1], $\mathcal{A}_n^t$ is determinizable, and hence t'-determinizable for every $t' \geq 1$. Assume by way of contradiction that there is a DWFA $\mathcal{A}'$ with less than 2^n states that t'-approximates $\mathcal{A}_n^t$, for some $t' < t$. For a word $w \in \{a,b\}^*$, let $\mathbf{c}(w)$ be the sum of the initial weight and the weights of the transitions of the single run of $\mathcal{A}'$ on w.

Consider a word $w \in \{a,b\}^+$. Recall that $cost(\mathcal{A}, w) \leq t$. Since $\mathcal{A}'$ t'-approximates $\mathcal{A}$, it follows that $cost(\mathcal{A}', w) \leq t' \cdot t$. Since all the weights in $\mathcal{A}'$ are non-negative[4], then $\mathbf{c}(w) \leq cost(\mathcal{A}', w)$. Hence, $\mathbf{c}(w) \leq t' \cdot t$ for all $w \in \{a,b\}^+$. Since $\mathcal{A}'$ is finite, and thus involves only finitely many weights, this implies that there is some cost $c \leq t' \cdot t$ such that $\mathbf{c}(w) \leq c$ for all words $w \in \{a,b\}^+$ and there is at least one word $u \in \{a,b\}^+$ for which $\mathbf{c}(u) = c$. Note that, by our choice of u, for all $v \in \{a,b\}^*$ we have that $\mathbf{c}(uv) = \mathbf{c}(u)$.

[4]Using similar but slightly more complex arguments one can show an exponential blow-up also in the case where negative weights are allowed.

Let $x = x_1 \dots x_n$ and $y = y_1 \dots y_n$ be two different words in $\{a, b\}^n$ such that $\mathcal{A}'$ reaches the same state after reading ux and uy. Since $\mathcal{A}'$ has less than 2^n states, such different x and y exist. Let $1 \leq i \leq n$ be such that $x_i \neq y_i$. Without loss of generality, let $x_i = a$ and $y_i = b$. Finally, let $z = a^{i-1}$. Note that $uxz \in L_n$ whereas $uyz \notin L_n$. Also, $\mathcal{A}'$ reaches the same state q after reading uxz and uyz. Since $cost(\mathcal{A}', uxz) \leq t'$, we have $\mathbf{c}(uxz) + f(q) \leq t'$. Also, since $cost(\mathcal{A}', uyz) \geq t$, we have that $\mathbf{c}(uyz) + f(q) \geq t$. Recall that by our choice of u, we have that $\mathbf{c}(u) = \mathbf{c}(uxz) = \mathbf{c}(uyz)$. If follows that $t \leq \mathbf{c}(u) + f(q) \leq t'$, contradicting the assumption that $t > t'$. ■

Before we turn to describe our t-determinization algorithm, let us recall *MDet* – Mohri's determinization algorithm [1]. The algorithm *MDet* is based on the subset construction for determinization of NFAs [16]. There, each state s of the deterministic automaton is associated with a set of states of $\mathcal{A}$, and the intuition is that the single run of the deterministic automaton reaches s iff $\mathcal{A}$ has a run that reaches q for exactly all $q \in s$. This general intuition holds also for *MDet*, except that now the different states in s may have been reached using runs with different costs, whereas the single run to s has a single cost. The construction of the deterministic automaton $\mathcal{A}_M$ makes sure that the cost of the single run is the minimal cost to some state in s. In order to achieve this, the states in $\mathcal{A}_M$ are attributed by additional information, and each state s is a set of pairs $\langle q, x\rangle$, where $q \in Q$ is a state in the input WFA, and $x \in \mathbb{R}^{\geq 0}$ is the *residual weight* of q in s. Intuitively, the residual weight of q in s is the difference between the cost of the minimal run to some state in s and the cost of the minimal run to q. This weight has to be taken into account if q is chosen to be the state from which $\mathcal{A}_M$ proceeds from s. In more detail, let s' be the set of states reachable from s by the letter a. There may be different weights on transitions from different states in s to their a-successors in s'. In order to determine the weight c of the single transition from s to s' in $\mathcal{A}_M$, we calculate for every state q' in s' a value $v_{q'}$. This value is the sum of the residual weight of the state $q \in s$ that is the origin of the a-transition to q', and the weight of the a-transition from q to q' (in case there are several such origins, we consider the one that minimizes this sum). The weight c in then set to the minimal $v_{q'}$ over all $q' \in s'$, and the residual weight of each state $q' \in s'$ is set to $v_{q'} - c$.

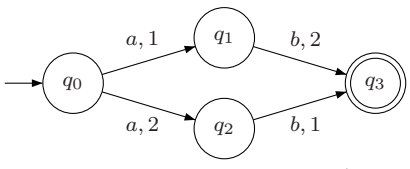

Fig. 5. The original WFA $\mathcal{A}$.

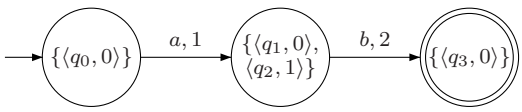

Fig. 6. The result $\mathcal{D}$ of Mohri's determinization algorithm.

In Figures 5 and 6 we show an example for applying *MDet*. From the state q_0 there are a-transitions to both q_1 and q_2. Thus, the set of states in the a-successor of the initial state of $\mathcal{D}$ consists of q_1 and q_2. Since $v_{q_1} = 1$ while $v_{q_2} = 2$, the transition in $\mathcal{D}$ gets the weight $v_{q_1} = 1$, and q_2 gets the residual weight of $2-1 = 1$, indicating that if the run continues from q_2, then 1 should be added to the cost accumulated in the transitions taken so far. The b-transition to the accepting state of $\mathcal{D}$ gets the weight of 2, since no matter which origin of the b-transition to q_3 we consider, we get $v_{q_3} = 2$ (residual weight 0 + weight 2 for the transition from q_1, and residual weight 1 + weight 1 for the transition from q_2). Note that if the weight on the transition from q_2 to q_3 had been 0, then the weight of the transition to the accepting state in $\mathcal{D}$ would have changed to 1, since in this case, considering the origin q_2, we would have got $v_{q_3} = 1 + 0 = 1$.

If we try to apply *MDet* to the non-determinizable WFA in Figure 1, *MDet* would generate infinitely many states with the subset $\{q_1, q_2\}$, as every time we take the b-transition from this subset to itself, the residual weight of q_2 increases by 1 and thus, a new state should be created.

One may be tempted to try overcoming this situation by multiplying the weights on a subset of the transitions by a factor bounded by t. Indeed, if we multiply the weight of the b-transition from q_1 to itself by 2, then the WFA we get can be determinized. However, as described in Section I, there are cases in which this attempt does not succeed. Apparently, the single residue maintained by *MDet* is not sufficiently informative to deal with this problem.

Our algorithm $tDet$ copes with this problem by associating each state of the WFA with a range of residual weights rather than with a single residual weight. This enables the unification of two states with the same subset even when their residual weights are not equal, as long as the unification does not result in a weight that is out of the allowed range. The ranges are simple, in the sense that they are defined by means of upper and lower bounds.

Given a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f\rangle$, $tDet$ constructs a DWFA $\mathcal{A}' = \langle \Sigma, P, \Delta', c', p_0, F', i', f'\rangle$ as follows. The state space of $\mathcal{A}'$ is $P \subseteq 2^{Q \times \mathbb{R} \times \mathbb{R}}$. Thus, every state $p \in P$ is a set of triples $\langle q, l_q, u_q\rangle$. The set of the *underlying states* of p, denoted $states(p)$, is the set $\{q : \langle q, l_q, u_q\rangle \in p\}$. Each one of the underlying states q, has its own *residual range* $[l_q, u_q]$ in p, defined by its *lower bound*, l_q, and *upper bound*, u_q. When p is not clear from the context, we use l_q^p and u_q^p.

The idea is that since the cost $cost(\mathcal{A}', w)$ assigned by $\mathcal{A}'$ to a word w should be between $cost(\mathcal{A}, w)$ and $t \cdot cost(\mathcal{A}, w)$, we should store for every underlying state q of p the minimal residual weight l_q that should be added to the cost of the run that proceeds from q in order to make sure that its cost is at least the cost of the cheapest corresponding run in $\mathcal{A}$, and the maximal residual weight u_q that may be added to the cost of the run that proceeds from q without causing its cost to exceed t times the cost of the cheapest corresponding run in $\mathcal{A}$. Accordingly, every residual weight within this range may be used, without violating the approximation. Therefore, in case the algorithm is about to create a new state p', and there

already exists a state r such that $states(r) = states(p') = S$, and for every $q \in S$ the residual range of q in r, $[l_q^r, u_q^r]$, is contained in the residual range of q in p', $[l_q^{p'}, u_q^{p'}]$ (that is, $l_q^{p'} \leq l_q^r \leq u_q^r \leq u_q^{p'}$), then the algorithm does not create p', but uses r instead. In this case we say that r *refines* p'.

procedure $tDet(\mathcal{A}, t)$
1 $P := \emptyset$; $F' := \emptyset$; $\mathcal{Q} := \emptyset$;
2 $i' := t \cdot \min\{i(q_0) : q_0 \in Q_0\}$;
3 $p_0 := \{\langle q_0, i(q_0) - i', t \cdot i(q_0) - i' \rangle : q_0 \in Q_0\}$;
4 $Enqueue(\mathcal{Q}, p_0)$;
5 **while** $\mathcal{Q} \neq \emptyset$ **do**
6 $p := Dequeue(\mathcal{Q})$; $P := P \cup \{p\}$;
7 **if** $states(p) \cap F \neq \emptyset$ **then**
8 $F' := F' \cup \{p\}$;
9 $f'(p) := \min_{q \in states(p) \cap F}\{l_q + f(q)\}$;[5]
10 **for each** $\{a \in \Sigma : \delta(states(p), a) \neq \emptyset\}$ **do**
11 $c' := \min\{u_q + t \cdot c(d) :$
$\langle q, l_q, u_q \rangle \in p$ and $d = (q, a, q') \in \Delta\}$;
12 $p' := \bigcup_{q' \in \delta(states(p),a)}\{\langle q',$
$\min\{l_q + c(d) - c' :$
$\langle q, l_q, u_q \rangle \in p$ and $d = \langle q, a, q' \rangle \in \Delta\}$,
$\min\{u_q + t \cdot c(d) - c' :$
$\langle q, l_q, u_q \rangle \in p$ and $d = \langle q, a, q' \rangle \in \Delta\}\rangle\}$;
13 **if** there is $r \in \mathcal{Q} \cup P$ such that r refines p' **then**
14 $d' := \langle p, a, r \rangle$; $c'(d') := c'$;
15 $\Delta' := \Delta' \cup d'$;
16 **else**
17 $d' := \langle p, a, p' \rangle$; $c'(d') := c'$;
18 $\Delta' := \Delta' \cup d'$; *Enqueue*$(\mathcal{Q}, p')$;

Fig. 7. The t-determinization algorithm.

In Figure 7 we describe $tDet$ in pseudo-code. The resulting DWFA $\mathcal{A}'$ is constructed on the fly, using a queue of states $\mathcal{Q}$. That is, initially, the initial weight i' is calculated[6], and the initial state p_0 is created and enqueued into $\mathcal{Q}$. Then, while $\mathcal{Q}$ is not empty, at each stage, one state p is being dequeued from $\mathcal{Q}$, processed, and added to P. Processing a state p includes two steps. First, if there exists at least one accepting state of $\mathcal{A}$ that belongs to $states(p)$ then p is defined as accepting, and its final weight is defined. Then, the algorithm calculates for every letter $a \in \Sigma$ the state $p' = \delta'(p, a)$ and the weight of the a-transition from p to p'. If there already exists a state $r \in P$ that refines p', then r is defined as the target of the a-transition from p, and p' is not created. In this case we say that the a-transition from p to r is *red*. Otherwise, p' is created and enqueued into $\mathcal{Q}$. In this case we say that the a-transition from p to p' is *green*. Note that because of the use of a queue, the DWFA $\mathcal{A}'$ is constructed in a BFS manner. Thus, states that are reachable from the initial state by shorter words are generated and processed before states that are reachable from the initial state by longer words.

[5]We could use u_q and $t \cdot f(q)$ instead l_q and $f(q)$, to be consistent about multiplying all weights by t, but we prefer to keep the cost as tight as possible.

[6]Since $\mathcal{A}'$ is deterministic, it has a single initial state. Accordingly, we refer to i' as a single value rather than a function.

Example 2. *The WFA in Figure 8 is the result of our t-determinization algorithm, applied with $t = 2$ on the non-determinizable WFA in Figure 1.*

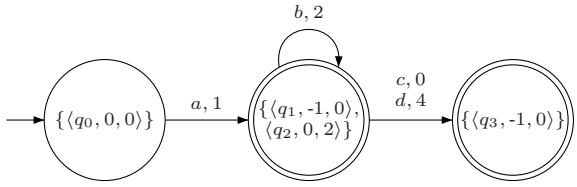

Fig. 8. The 2-determinization of the WFA in Figure 1.

A simple analysis of the algorithm $tDet$ in Figure 7 yields the following two lemmas:

Lemma 3. *For every state $p \in P$, we have that* (i) *for every $\langle q, l_q, u_q \rangle \in p$ it holds that $u_q \geq 0$; and* (ii) *there exists $\langle q, l_q, u_q \rangle \in p$ such that $u_q = 0$.*

Lemma 4. *The weight $c'(d')$ of every transition $d' \in \Delta'$ is non-negative.*

Theorem 5. *If the determinization algorithm terminates, then the resulting DWFA $\mathcal{A}'$ t-approximates the given WFA $\mathcal{A}$.*

Proof: Recall that for a word $w \in \Sigma^*$, and states $q, q' \in Q$, the *partial cost* of w from q to q' in $\mathcal{A}$, $\theta(q, w, q')$, is the cost of the cheapest partial run of $\mathcal{A}$ on w from q to q'. In addition, let $\theta(Q_0, w, q) = \min_{q_o \in Q_0}[i(q_0) + \theta(q_0, w, q)]$. Also, for $p \in P$, let $\theta'(p, w)$ denote the cost of the unique path labeled by w starting at p in $\mathcal{A}'$. We show that for all $w \in \Sigma^*$, the state $\delta'(p_0, w) \in P$ satisfies the following.

$$states(\delta'(p_0, w)) = \delta(Q_0, w), \tag{1}$$

and for every $\langle q, l_q, u_q \rangle \in \delta'(p_0, w)$, we have

$$\theta(Q_0, w, q) - [i' + \theta'(p_0, w)] \leq l_q \leq u_q \leq t \cdot \theta(Q_0, w, q) - [i' + \theta'(p_0, w)]. \tag{2}$$

We prove both claims by an induction on $|w|$. The base case is when $w = \varepsilon$. In this case, it is clear from the algorithm (line 3) that indeed $states(p_0) = Q_0 = \delta(Q_0, \varepsilon)$. Since for a state $q \in Q_0$, the only path from any state in Q_0 to q while reading ε is q itself, we have

$$\begin{aligned}\theta(Q_0, \varepsilon, q) \; - \; & [i' + \theta'(p_0, \varepsilon)] = \\ & = i(q) + \theta(q, \varepsilon, q) - [i' + \theta'(p_0, \varepsilon)] \\ & = i(q) - i' \; [= l_q] \\ & \leq t \cdot i(q) - i' \; [= u_q] \\ & = t \cdot \theta(Q_0, \varepsilon, q) - [i' + \theta'(p_0, \varepsilon)].\end{aligned}$$

The induction step for (1) is simple. Consider a word $w = ua$, where $u \in \Sigma^*$ and $a \in \Sigma$. By the induction hypothesis, we have

$$states(\delta'(p_0, u)) = \delta(Q_0, u). \tag{*}$$

Consider a state $p \in P$. From the algorithm (line 12), we have

$$states(\delta'(p,a)) = \delta(states(p),a). \qquad (**)$$

Thus, we have

$$\begin{aligned} states(\delta'(p_0,w)) &= states(\delta'(\delta'(p_0,u),a)) \\ &= \delta(states(\delta'(p_0,u)),a) \quad (3) \\ &= \delta(\delta(Q_0,u),a) = \delta(Q_0,w), \quad (4) \end{aligned}$$

where (3) holds due to (**) and (4) holds due to (*).

The induction step for (2) is more involved. First, we show that for every $\langle q', l_{q'}, u_{q'}\rangle \in \delta'(p_0,w)$ we have $l_{q'} \leq u_{q'}$. For $|w| = k > 0$ and $p' = \delta'(p_0,w)$, if p' can be reached from p_0 while reading a word shorter than w, then, by the induction hypothesis, we are done. Otherwise, let $p \in P$ be the state from which p' is primarily reached, i.e., $p = \delta'(p_0,u)$ for $|u| = k-1$, and the state p' is generated while processing the state p with a letter a (line 12). Thus, $\langle p,a,p'\rangle \in \Delta'$. Let $c' = c'(\langle p,a,p'\rangle)$. For a state $q' \in states(p')$, let $\overline{q} \in states(p)$ be a state for which $\langle \overline{q},a,q'\rangle \in \Delta$ and for which $\overline{c} = c(\langle \overline{q},a,q'\rangle)$ is such that $u_{\overline{q}} + t\cdot\overline{c} = \min\{u_q + t\cdot c(d) : \langle q,l_q,u_q\rangle \in p$ and $d = (q,a,q') \in \Delta\}$. Then, by Lemmas 3 and 4, we have

$$\begin{aligned} l_{q'} &= \min\{l_q + c(d) - c' : \\ &\qquad \langle q,l_q,u_q\rangle \in p \text{ and } d = \langle q,a,q'\rangle \in \Delta\} \\ &\leq l_{\overline{q}} + \overline{c} - c' \\ &\leq l_{\overline{q}} + t\cdot\overline{c} - c' \\ &\leq u_{\overline{q}} + t\cdot\overline{c} - c' \quad \text{(by the induction hypothesis)} \\ &= \min\{u_q + t\cdot c(d) - c' : \\ &\qquad \langle q,l_q,u_q\rangle \in p \text{ and } d = \langle q,a,q'\rangle \in \Delta\} = u_{q'}. \end{aligned}$$

To complete the proof of (2), we show that for every $\langle q', l_{q'}, u_{q'}\rangle \in \delta'(p_0,w)$ we have

$$\theta(Q_0,w,q') - [i' + \theta'(p_0,w)] \leq l_{q'}$$

and

$$u_{q'} \leq t\cdot\theta(Q_0,w,q') - [i' + \theta'(p_0,w)].$$

Let $w = u\cdot a$, for $u \in \Sigma^*$ and $a \in \Sigma$, and let $p = \delta'(p_0,u)$. For $q' \in \delta(Q_0,w)$ we have

$$\begin{aligned} &\theta(Q_0,w,q') - [i' + \theta'(p_0,w)] = \\ &\quad = \min_{q\in\delta(Q_0,u)}\{\theta(Q_0,u,q) + c(\langle q,a,q'\rangle)\} \\ &\qquad -[i' + \theta'(p_0,u) + c'(\langle p,a,p'\rangle)] \\ &\quad \leq \min_{q\in\delta(Q_0,u)}\{l_q + c(\langle q,a,q'\rangle)\} - c'(\langle p,a,p'\rangle) \quad (5) \\ &\quad \leq l_{q'}, \quad (6) \end{aligned}$$

and

$$\begin{aligned} u_{q'} &\leq \min_{q\in\delta(Q_0,u)}\{u_q + t\cdot c(\langle q,a,q'\rangle)\} - c'(\langle p,a,p'\rangle) \quad (7) \\ &\leq \min_{q\in\delta(Q_0,u)}\{t\cdot\theta(Q_0,u,q) - [i' + \theta'(p_0,u)] \quad (8) \\ &\qquad\qquad +t\cdot c(\langle q,a,q'\rangle)\} \\ &\qquad -c'(\langle p,a,p'\rangle) \\ &= t\cdot\theta(Q_0,w,q') - [i' + \theta'(p_0,w)], \end{aligned}$$

where inequalities (5) and (8) hold due to the induction hypothesis, and inequalities (6) and (7) hold due to the possibility that a state is replaced by a state that refines it.

By (1) and the definition of F', a word w is accepted by $\mathcal{A}$ iff w is accepted by $\mathcal{A}'$.
Let w be a word accepted by both $\mathcal{A}$ and $\mathcal{A}'$. By applying (2) on l_q appearing in line 9 of the algorithm, we get

$$\begin{aligned} &\min_{q\in\delta(Q_0,w)\cap F}\{\theta(Q_0,w,q) + f(q)\} - [i' + \theta'(p_0,w)] \\ &\leq f'(\delta'(p_0,w)) \\ &\leq \min_{q\in\delta(Q_0,w)\cap F}\{t\cdot\theta(Q_0,w,q) + f(q)\} - [i' + \theta'(p_0,w)]. \end{aligned}$$

Thus,

$$\begin{aligned} &cost(\mathcal{A},w) - i' - \theta'(p_0,w) \\ &\qquad \leq f'(\delta'(p_0,w)) \\ &\qquad \leq t\cdot cost(\mathcal{A},w) - i' - \theta'(p_0,w). \end{aligned}$$

From the first inequality we get

$$cost(\mathcal{A},w) \leq i' + \theta'(p_0,w) + f'(\delta'(p_0,w)) \qquad (9)$$

From the second inequality we get

$$i' + \theta'(p_0,w) + f'(\delta'(p_0,w)) \leq t\cdot cost(\mathcal{A},w). \qquad (10)$$

Since the right hand of (9) and the left hand of (10) both equal $cost(\mathcal{A}',w)$, we have

$$cost(\mathcal{A},w) \leq cost(\mathcal{A}',w) \leq t\cdot cost(\mathcal{A},w).$$

■

IV. The t-Twins Property

In this section we define and study the *t-twins property* of WFA. Consider a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f\rangle$. Two states, p and q, of $\mathcal{A}$ are *t-twins* if for all $u,v \in \Sigma^*$ such that $p,q \in \delta(Q_0,u)$, $p \in \delta(p,v)$, and $q \in \delta(q,v)$, it holds that $\theta(p,v,p) \leq t\cdot\theta(q,v,q)$. The WFA $\mathcal{A}$ has the *t-twins property* if all pairs $p,q \in Q$ are t-twins. Thus, the t-twins property bounds by t the ratio between the costs of traversing cycles that can be traversed reading the same word. Note that the twins property is simply the t-twins property for $t = 1$. Also note that if $\mathcal{A}$ does not contain cycles of weight 0, then it satisfies the t-twins property, for some t. When $\mathcal{A}$ does contain such cycles, it may not satisfy the t-twins property, for all t.

We now prove that for WFAs with rational weights and for a rational approximation factor t, if the t-twins property holds then our t-determinization algorithm always terminates.

Theorem 6. *Consider a WFA $\mathcal{A}$ in which the weights are in $\mathbb{Q}^{\geq 0}$, and an approximation factor $t \in \mathbb{Q}, t \geq 1$. If $\mathcal{A}$ satisfies the t-twins property, then $tDet(\mathcal{A},t)$ terminates.*

Proof: Let $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f\rangle$, and let $n = |Q|$. First, observe that if all the weights appearing in $\mathcal{A}$ are rational, we can multiply them all by a common denominator, and thus assume that all the weights in $\mathcal{A}$ are natural numbers.

Assume by way of contradiction that $\mathcal{A}$ satisfies the t-twins property but that $tDet(\mathcal{A},t)$ does not terminate. Thus,

$tDet(\mathcal{A}, t)$ generates infinitely many states in the process of constructing the DWFA $\mathcal{A}'$. Observe that every new state p' that is added to $\mathcal{A}'$ (line 18 of the algorithm) has an incoming green edge (line 17). Hence, by a simple induction on the iteration in which a state is added to $\mathcal{A}'$, we can show that every state in $\mathcal{A}'$ that is reachable from the initial state p_0 is also reachable by green edges only. Since every state in $\mathcal{A}'$ has at most one outgoing edge for every letter in the finite alphabet Σ, then, by König's Lemma, there is an infinite path $\pi = p_0 p_1 \ldots$ of distinct states (i.e., $j \neq k \implies p_j \neq p_k$) in $\mathcal{A}'$, that, by the above observation, uses only green edges. Since $\mathcal{A}$ has only finitely many states, it follows that there is a set $S = \{q_0, \ldots, q_m\} \subseteq Q$ such that the set of indices $J_0 = \{j \in \mathbb{N} : states(p_j) = S\}$ is infinite. For every state $q \in S$ and every index $j \in J_0$, let $l_{q,j}$ denote the lower residue of q in p_j, and let $u_{q,j}$ denote the upper residue of q in p_j. We claim that there must be a state $p \in S$ such that the sequence of lower residues $\{l_{p,j} : j \in J_0\}$ is unbounded from above.

In order to prove the above claim, assume by way of contradiction that there is some constant $b \in \mathbb{N}$ such that $l_{q,j} \leq b$ for all $q \in S$ and all $j \in J_0$. By Lemma 3, we have that $u_{q,j} \geq 0$ for all $q \in S$ and all $j \in J_0$. Recall that all the weights in $\mathcal{A}$ are in $\mathbb{N}$, and thus, all the lower and upper residues in $\mathcal{A}'$ are in $\mathbb{Z}$. Every infinite sequence of integers that is bounded from above (below) either tends towards $-\infty$ (∞, respectively), or it has some integer that repeats infinitely often. We derive a series of infinite sequences of indices $J_0 \supseteq J_1 \ldots \supseteq J_{2m+2}$ (recall that $S = \{q_0, \ldots, q_m\}$) as follows: for $0 \leq k \leq m$, if the sequence $\{l_{q_k,j} : j \in J_{2k}\}$ tends to $-\infty$, then we let $J_{2k+1} = J_{2k}$; and if there is $\ell_k \in \mathbb{Z}$ that appears in $\{l_{q_k,j} : j \in J_{2k}\}$ infinitely often, then we let $J_{2k+1} = \{j \in J_{2k} : l_{q_k,j} = \ell_k\}$. Similarly, if the sequence $\{u_{q_k,j} : j \in J_{2k+1}\}$ tends to ∞, then we let $J_{2k+2} = J_{2k+1}$, and otherwise, we let $J_{2k+2} = \{j \in J_{2k+1} : u_{q_k,j} = \mathsf{u}_k\}$, for $\mathsf{u}_k \in \mathbb{Z}$ that appears infinitely often in $\{u_{q_k,j} : j \in J_{2k+1}\}$.

Consider now the subsequence π^{2m+2}, of states of π, defined by $\pi^{2m+2} = \{p_j : j \in J_{2m+2}\}$, and let j_0 be the minimal index in J_{2m+2}. It is easy to see that, by our construction, for every $q_k \in S$ we have that either the lower (upper) residue of q_k is the same for all states in π^{2m+2} or it tends to $-\infty$ (∞, respectively). It follows that for all large enough $j \in J_{2m+2}$ and for all $q_k \in S$, it holds that $l_{q_k,j} \leq l_{q_k,j_0}$ and $u_{q_k,j} \geq u_{q_k,j_0}$. In other words, there is an (infinite) suffix of π^{2m+2} all of whose states are refined by p_{i_0}. This is, however, impossible, as $tDet(\mathcal{A}, t)$ (line 13 of the algorithm) never adds to $\mathcal{A}'$ a state that is refined by a previously added state, and obviously at the time that p_{i_0} was added to $\mathcal{A}'$ only finitely many states were already present in $\mathcal{A}'$. This proves our claim that there is a state $p \in S$ such that the sequence of lower residues $\{l_{p,j} : j \in J_0\}$ is unbounded. We can thus take an infinite subsequence $\tilde{J} \subseteq J_0$ for which the lower residues of p monotonically increase towards ∞. We complete the proof of the theorem by showing that the fact that the sequence of lower residues $\{l_{p,j} : j \in \tilde{J}\}$ is monotonically increasing towards infinity implies that $\mathcal{A}$ does not satisfy the t-twins property. By Lemma 3 and the fact that $\mathcal{A}$ has only finitely many states, we have that there is a state $q \in S$ such that $u_{q,j} = 0$ for every j in some infinite subsequence $J' \subseteq \tilde{J}$. Consider now the subsequence π' of states of π defined by $\pi' = \{p_j : j \in J'\}$. Given a word $w \in \Sigma^*$, let $runs(w, p)$, and $runs(w, q)$ be the sets of all partial runs of $\mathcal{A}$ on w that reach p and q, respectively. Let $x = \max\{c(r) - t \cdot c(r') : w \in \Sigma^*, |w| \leq n^2, r \in runs(w, p), r' \in runs(w, q)\}$. I.e., x is the maximal value that the expression $c(r) - t \cdot c(r')$ attains when r and r' range over all possible partial runs of $\mathcal{A}$ (that respectively reach p and q) on words of length at most n^2. Observe that, by our choice of π, every state p_j in π' is reachable from the initial state p_0 by reading some word w^j using only green edges. Hence, by the proof of Theorem 5, for every $j \in J'$ we have $\theta(Q_0, w^j, p) - i' - \theta'(p_0, w^j) = l_{p,j}$, and $t \cdot \theta(Q_0, w^j, q) - i' - \theta'(p_0, w^j) = u_{q,j}$.

By subtracting the second equation from the first, and recalling that $u_{q,j} = 0$, we get that $\theta(Q_0, w^j, p) - t \cdot \theta(Q_0, w^j, q) = l_{p,j}$ for every $j \in J'$. Since J' is an infinite subsequence of $\tilde{J}$, the lower residues of p in π' tend towards infinity, and thus, the last equation implies that there is an index k in J' such that $\theta(Q_0, w^k, p) - t \cdot \theta(Q_0, w^k, q) > x$. Let z be the length of w^k, and let $r(w^k, p) = r_0, r_1 \ldots r_z$ be a partial run of $\mathcal{A}$ on w^k that ends in p and costs $\theta(Q_0, w^k, p)$. Similarly, let $r(w^k, q) = r'_0, r'_1 \ldots r'_z$ be a partial run of $\mathcal{A}$ on w^k that ends in q and costs $\theta(Q_0, w^k, q)$.

Observe that by our choice of k, we have that $z > n^2$, and thus there are two indices $0 \leq i < j \leq z$ such that $r_i = r_j$ and $r'_i = r'_j$. Consider the word $u = w_1^k \ldots w_i^k \cdot w_{j+1}^k \ldots w_z^k$. It is easy to see that the run $r(u, p) = r_0 \ldots r_i r_{j+1} \ldots r_z$, obtained by removing the loop $r_i \ldots r_j$ from $r(w^k, p)$, is a run of $\mathcal{A}$ on u that ends in p. Similarly, the partial run $r(u, q) = r'_0 \ldots r'_i r'_{j+1} \ldots r'_z$, obtained by removing the loop $r'_i \ldots r'_j$ from $r(w^k, q)$, is a partial run of $\mathcal{A}$ on u that ends in q. Let $\mathsf{d}_p = c(r(w^k, p)) - c(r(u, p))$ and $\mathsf{d}_q = c(r(w^k, q)) - c(r(u, q))$ be the respective differences in the costs of the above partial runs on u and w^k. Observe that d_p is the cost of looping in $r_i \ldots r_j$ whereas d_q is the cost of looping in $r'_i \ldots r'_j$. Since, by our assumption, $\mathcal{A}$ satisfies the t-twins property, we have that $\mathsf{d}_p \leq t \cdot \mathsf{d}_q$, and thus, $c(r(w^k, p)) - c(r(u, p)) \leq t \cdot [c(r(w^k, q)) - c(r(u, q))]$. Rearranging the last inequality we get that $c(r(u, p)) - t \cdot c(r(u, q)) \geq c(r(w^k, p)) - t \cdot c(r(w^k, q))$. Hence, by removing a synchronized loop from $r(w^k, p)$ and $r(w^k, q)$, the difference between the cost of the remaining run to p and t times the cost of the remaining run to q does not decrease. It follows that by repeatedly removing such synchronized loops from $r(w^k, p)$ and $r(w^k, q)$ we can obtain a word v of length at most n^2 such that the partial runs $r(v, p)$ and $r(v, q)$ satisfy $c(r(v, p)) - t \cdot c(r(v, q)) \geq c(r(w^k, p)) - t \cdot c(r(w^k, q))$. Recall that we chose $r(w^k, p)$ and $r(w^k, q)$ such that $c(r(w^k, p)) = \theta(Q_0, w^k, p)$ and $c(r(w^k, q)) = \theta(Q_0, w^k, q)$, and that by our choice of k we have that $\theta(Q_0, w^k, p) - t \cdot \theta(Q_0, w^k, q) > x$. Combining the last four (in)equalities we get that $c(r(v, p)) - t \cdot c(r(v, q)) > x$, which is a contradiction since, by the definition of x and the fact that $|v| \leq n^2$, we have that

$c(r(v,p)) - t \cdot c(r(v,q)) \leq x$. ■

By Theorems 5 and 6, the algorithm $tDet$ successfully t-determinizes all WFAs with rational weights that satisfy the t-twins property, for a rational $t \geq 1$. Note that the assumption about the weights being rational enabled us to use the well order on the natural numbers in the proof of Theorem 6. We were not able to prove termination for the general tropical semiring, and we leave open the problem whether $tDet$ may not terminate for WFAs that satisfy the t-twins properties but have irrational weights.

We now prove that the t-twins property captures a significant subclass of t-determinizable WFAs. In particular, as has been the case with determinization, the t-twins property characterizes exactly the subclass of trim and unambiguous t-determinizable WFAs.

Theorem 7. *Consider a WFA $\mathcal{A}$ in which the weights are in $\mathbb{Q}^{\geq 0}$, and an approximation factor $t \in \mathbb{Q}, t \geq 1$. If $\mathcal{A}$ is trim and unambiguous, then $\mathcal{A}$ is t-determinizable iff $\mathcal{A}$ satisfies the t-twins property.*

Proof: Let $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f \rangle$. By Theorems 5 and 6, if $\mathcal{A}$ satisfies the t-twins property then it is t-determinizable. It remains to show that if $\mathcal{A}$ is trim, unambiguous, and t-determinizable then it satisfies the t-twins property.

Assume by way of contradiction that $\mathcal{A}$ is trim, unambiguous, and t-determinizable, but does not satisfy the t-twins property. Hence, there are two states $p, q \in Q$, and two words $u, w \in \Sigma^*$ such that $\mathcal{A}$ can reach both p and q by reading u, and it can loop from p to itself, as well as from q to itself, while reading w. Furthermore, the cost of looping on p is more than t times the cost of looping on q. Formally, let $r(Q_0, u, p)$ and $r(Q_0, u, q)$ be partial runs of $\mathcal{A}$ on u that reach (from some initial states) p and q respectively, and let $r(p, w, p)$ and $r(q, w, q)$ be partial runs of $\mathcal{A}$ on w from p to itself and from q to itself, respectively. Then, $c(r(p, w, p)) > t \cdot c(r(q, w, q))$. Since $\mathcal{A}$ is trim, there are words $v, \tilde{v} \in \Sigma^*$ such that there is a partial run $r(p, v)$ of $\mathcal{A}$ from p to some accepting state, and a partial run $r(q, \tilde{v})$ of $\mathcal{A}$ from q to some (maybe different) accepting state. Also, since $\mathcal{A}$ is unambiguous, then for every $j \geq 0$, the run obtained by following $r(Q_0, u, p)$ then looping j times along $r(p, w, p)$ and finally following $r(p, v)$, is the only accepting run of $\mathcal{A}$ on the word $u \cdot w^j \cdot v$. Similarly, the run obtained by following $r(Q_0, u, q)$ then looping j times along $r(q, w, q)$ and finally following $r(q, \tilde{v})$ is the only accepting run of $\mathcal{A}$ on the word $u \cdot w^j \cdot \tilde{v}$. It follows that $cost(\mathcal{A}, u \cdot w^j \cdot v) = c(r(Q_0, u, p)) + j \cdot c(r(p, w, p)) + c(r(p, v)) + x$, and $cost(\mathcal{A}, u \cdot w^j \cdot \tilde{v}) = c(r(Q_0, u, q)) + j \cdot c(r(q, w, q)) + c(r(q, \tilde{v})) + y$, where the constants x and y (which represent the sums of the initial and final costs) are independent of j.

Recall that $\mathcal{A}$ is t-determinizable. Let $\mathcal{A}' = \langle \Sigma, Q', \Delta', c', Q'_0, F', i', f' \rangle$ be a DWFA that t-approximates $\mathcal{A}$. Since $\mathcal{A}'$ is finite, there is some state s of $\mathcal{A}'$, and an infinite sequence of indices $J \subseteq \mathbb{N}$, such that $\mathcal{A}'$ reaches s after reading $u \cdot w^j$ for every $j \in J$. Being deterministic, it follows that for every $j \in J$, the state s is the only state reachable in $\mathcal{A}'$ after reading $u \cdot w^j$, and that since $\mathcal{A}'$ t-approximates $\mathcal{A}$ it must accept both v and $\tilde{v}$ from s. Let $r'(u \cdot w^j)$ be the run of $\mathcal{A}'$ (from the initial state) on $u \cdot w^j$, and let $r'(s, v)$ and $r'(s, \tilde{v})$ be the partial runs of $\mathcal{A}'$ from s on v and $\tilde{v}$, respectively. We thus have that $cost(\mathcal{A}', u \cdot w^j \cdot v) = c'(r(u \cdot w^j)) + c'(r(s, v)) + x'$ and $cost(\mathcal{A}', u \cdot w^j \cdot \tilde{v}) = c'(r(u \cdot w^j)) + c'(r(s, \tilde{v})) + y'$, where the constants x' and y' (which represent the sums of the initial and final costs) are independent of j.

Now, since $\mathcal{A}'$ t-approximates $\mathcal{A}$, the cost of accepting a word in $\mathcal{A}'$ is at least the cost of accepting it in $\mathcal{A}$, and at most t times that cost. Thus, $c'(r(u \cdot w^j)) + c'(r(s, v)) + x' \geq c(r(Q_0, u, p)) + j \cdot c(r(p, w, p)) + c(r(p, v)) + x$, and $c'(r(u \cdot w^j)) + c'(r(s, \tilde{v})) + y' \leq t \cdot [c(r(Q_0, u, q)) + j \cdot c(r(q, w, q)) + c(r(q, \tilde{v})) + y]$. By rearranging and combining the last two inequalities, we get that for every $j \in J$, we have that $j \cdot c(r(p, w, p)) \leq t \cdot j \cdot c(r(q, w, q)) + a$, where a is a constant that is independent of j. Since J is infinite, the last inequality holds for j as large as we want, and thus $c(r(p, w, p)) \leq t \cdot c(r(q, w, q))$. This, however, contradicts our choice of p, q and w for which $c(r(p, w, p)) > t \cdot c(r(q, w, q))$, and we are done. ■

Deciding the t-twins Property

In [17], the authors presented an efficient polynomial algorithm for deciding whether a given trim and unambiguous WFA has the twins property (i.e., the t-twins property for $t = 1$). As we now show, extending this algorithm to handle the case $t \geq 1$ is not difficult.

Recall that $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F, i, f \rangle$ does not satisfy the t-twins property iff there are two states $p, q \in Q$, and two words $u, w \in \Sigma^*$ such that $\mathcal{A}$ can reach both p and q by reading u, and it can loop from p to itself, as well as from q to itself, while reading w. Furthermore, the cost of looping with w on p is more than t times the cost looping with w on q. A key observation is that if the t-twins property does not hold then its violation can be witnessed using a word w of length at most n^2.

The algorithm thus proceeds in two phases. In the first phase, it identifies all the pairs of states p, q that can be reached from the initial state using the same word u. Observe that this is a question about the non-emptiness of the intersection of regular languages (without weights), which can be easily solved in polynomial time. In the second phase, every such pair p, q is checked for the existence of violating loops, as follows: starting from p, unwind $\mathcal{A}$ for n^2 steps into a DAG G_p; similarly, starting from q, unwind $\mathcal{A}$ for n^2 steps into a DAG G_p; finally, construct the product DAG $G_p \times G_q$, as usual. I.e., $G_p \times G_q$ has a transition $\langle (x, x'), a, (y, y') \rangle$ iff there is a transition $\langle x, a, y \rangle$ in G_p, and a transition $\langle (x', a, y' \rangle$ in G_q. The weight of a transition $\langle (x, x'), a, (y, y') \rangle$ is $t \cdot c(\langle x, a, y \rangle) - c(\langle x', a, y' \rangle)$. Observe that if $\mathcal{A}$ is trim and unambiguous, then for every word w there is at most one path in $G_p \times G_q$ from (p, q) labeled by the letters of w, and its cost is exactly t times the cost of looping on p with w minus the cost of looping on q with w. Thus, a pair p, q witnesses a violation of the t-twins property iff there is a path of negative

cost in $G_p \times G_q$ from (p, q) to itself. The later can be efficiently checked by searching for the minimal-cost path from (p, q) back to itself (for example, using a topological sort).

Hence, we can conclude with the following:

Theorem 8. *It can be decided in polynomial time whether a trim and unambiguous WFA satisfies the t-twins property.*

When applied to ambiguous WFAs,[7] the algorithm above has a one-sided error and may miss WFAs that satisfy the t-twins property.

V. Discussion

We described a t-determinization algorithm for WFAs. We defined the t-twins property and showed that our construction successfully t-determinizes WFAs that satisfy the property, and that the property captures a large and natural subclass of t-determinizable WFAs. In particular, the t-twins property characterizes exactly t-determinizability for WFAs that are trim and unambiguous. We also described a polynomial-time algorithm for deciding the t-twins property.

An important open question regarding the determinization of weighted automata is the problem of deciding whether a given WFA is determinizable. This problem generalizes to the problem of deciding, given a WFA $\mathcal{A}$ and $t \geq 1$, whether $\mathcal{A}$ can be t-determinized, and thus also to the problem of finding the minimal t, if exists, for which a given WFA is t-determinizable.

Approximated determinization can be used not only for WFAs that are not determinizable but also for WFAs that are determinizable but whose exact determinization results in DWFAs that are too big. As we showed, the approximation may lead to a significant reduction in the state space. Such an approach is similar to the one used in approximation algorithms, where one settles for an approximated solution for complex optimization problems. In [10], we related the two approaches and described an automata-theoretic approach for the competitive analysis of online algorithms. The approach is based on modeling optimization problems by a WFA whose transitions correspond to actions of the algorithm. By relating the "unbounded look ahead" of optimal offline algorithms with nondeterminism, and relating the "no look ahead" of online algorithms with determinism, it is possible to solve problems about the competitive ratio of online algorithms, by reducing them to questions about approximated determinization of weighted automata. The framework in [10] had to restrict attention to DWFAs that can be obtained by pruning the transitions of the given WFA. Essentially, the WFA models the offline algorithm, which embodies all online algorithms, and its transitions correspond to requests handled by the algorithm. Having a t-determinization construction enables a simpler modeling of online algorithms, in which the correspondence between transitions of the WFA and actions of the algorithm follows from the alphabet of the WFA rather than from the requirement that the DWFA be obtained by pruning of transitions of the WFA.

[7] Note that it is easy to transform, in polynomial time, every WFA to an equivalent trim one.

Acknowledgments We thank Shir Peled for helpful discussions and the anonymous reviewers for their valuable detailed comments.

References

[1] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, pp. 269–311, 1997.
[2] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[3] A. Aho, "Algorithms for finding patterns in strings," *Handbook of Theoretical Computer Science*, pp. 255–300, 1990.
[4] M. Vardi, "Nontraditional applications of automata theory," in *Proc. 11th Symp. on Theoretical Aspects of Computer Science*, ser. Lecture Notes in Computer Science, vol. 789. Springer, 1994, pp. 575–597.
[5] J. Büchi, "On a decision method in restricted second order arithmetic," in *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*. Stanford University Press, 1962, pp. 1–12.
[6] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*, ser. EATCS Monographs on Theoretical Computer Science. Springer, 1986.
[7] A. Chakrabarti, K. Chatterjee, T. Henzinger, O. Kupferman, and R. Majumdar, "Verifying quantitative properties using bound functions," in *Proc. 13th Conf. on Correct Hardware Design and Verification Methods*, ser. Lecture Notes in Computer Science, vol. 3725. Springer, 2005, pp. 50–64.
[8] K. Chatterjee, L. Doyen, and T. Henzinger, "Quantative languages," in *Proc. 17th Annual Conf. of the European Association for Computer Science Logic*, ser. Lecture Notes in Computer Science, vol. 5213, 2008, pp. 385–400.
[9] C. Baier, N. Bertrand, and M. Grösser, "Probabilistic automata over infinite words: Expressiveness, efficiency, and decidability," in *Proc. 11th International Workshop on Descriptional Complexity of Formal Systems*, 2006, pp. 3 – 16.
[10] B. Aminof, O. Kupferman, and R. Lampert, "Reasoning about online algorithms with weighted automata," *ACM Transactions on Algorithms*, vol. 6, no. 2, 2010.
[11] K. Culik and J. Kari, "Digital images and formal languages," *Handbook of formal languages, vol. 3: beyond words*, pp. 599–616, 1997.
[12] M. Mohri, F. Pereira, and M. Riley., "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.
[13] D. Krob, "The equality problem for rational series with multiplicities in the tropical semiring is undecidable," *International Journal of Algebra and Computation*, vol. 4, no. 3, pp. 405–425, 1994.
[14] C. Allauzen and M. Mohri, "Efficient algorithms for testing the twins property," *Journal of Automata, Languages and Combinatorics*, vol. 8, no. 2, pp. 117–144, 2003.
[15] A. L. Buchsbaum, R. Giancarlo, and J. Westbrook, "An approximate determinization algorithm for weighted finite-state automata," *Algorithmica*, vol. 30, no. 4, pp. 503–526, 2001.
[16] M. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, pp. 115–125, 1959.
[17] A. L. Buchsbaum, R. Giancarlo, and J. Westbrook, "On the determinization of weighted finite automata," *SIAM J. Comput.*, vol. 30, no. 5, pp. 1502–1531, 2000.
[18] D. Kirsten and I. Mäurer, "On the determinization of weighted automata," *Journal of Automata, Languages and Combinatorics*, vol. 10, no. 2-3, pp. 287–312, 2005.
[19] P. Kenny, R. Hollan, V. Gupta, M. Lennig, P. Mermelstein, and D. O'Shaughnessy, "A*-admissible heuristics for rapid lexical access," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 1, pp. 49–58, 1993.
[20] R. Lacouture and R. D. Mori, "Lexical tree compression," in *2nd European Conference on Speech Communication and Technology*, vol. 2, 1991, pp. 581–584.
[21] S. Peled, "Private communications," 2010.